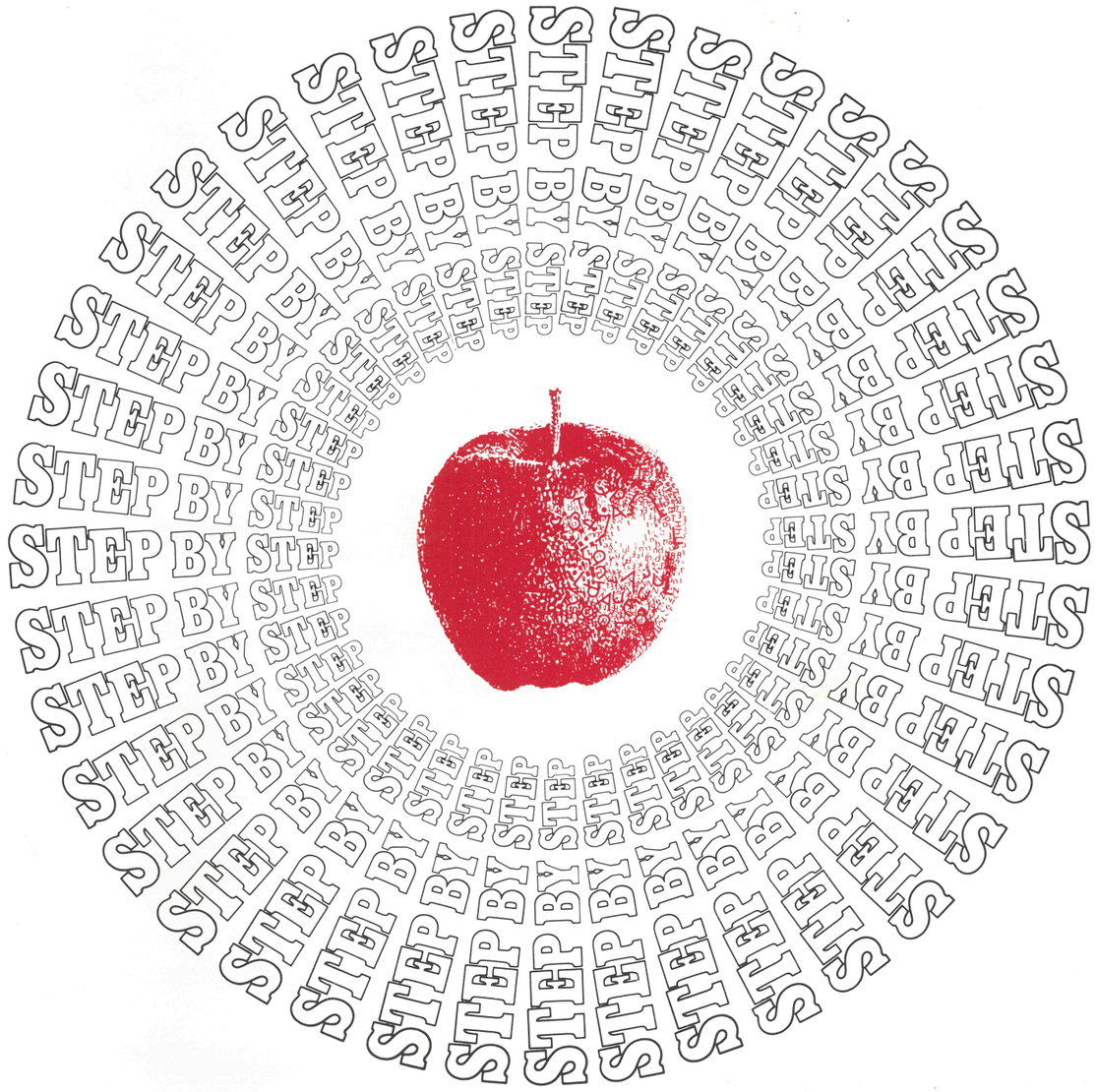
THE NEW STEP BY STEP

An Interactive Course in BASIC Programming for Beginners

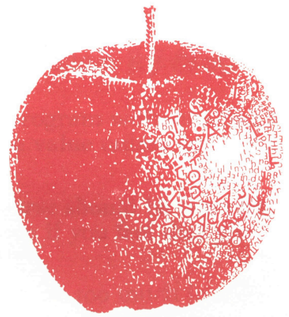


PDI

Program Design, Inc. 11 Idar Court Greenwich, Conn. 06830 203-661-8799

THE NEW STEP BY STEP

An Interactive Course in BASIC Programming for Beginners



PDI

Program Design, Inc. 11 Idar Court Greenwich, Conn. 06830 203-661-8799

Project Designer: John Victor

Programmers: Stephen Chmielewski
Kathleen Fortmeier
Lee Shepard

Editor: Regina Simon

Graphic Design: Howard Petlack

Copyright © 1981 by Program Design, Inc.
All rights reserved.

No part of this publication may be reproduced
in whole or in part, or stored in a retrieval system, or transmitted
in any form or by any means, electronic, mechanical,
photocopying or recording, without written permission of the publisher.
For information relating to permissions,
write to Program Design, Inc.,
11 Idar Court, Greenwich, CT 06830

Second Edition

Printed in the U.S.A.

What is The New Step By Step?

The New Step By Step is a course designed to teach beginners how to program in **BASIC**. This course is based on the original *Step By Step* developed in 1978. The original course took more than a year to develop using five different computers and using junior high school students, homemakers and other beginning programmers as test subjects.

Since that time, we have developed advanced techniques in presenting teaching material. We have added a voice cassette, animation and graphics in the teaching process.

Computer professionals tend to think of **BASIC** as a simple programming language that anyone can learn in a few hours of practice. The fact is that although **BASIC** may be *easier* to learn than assembly language or **FORTRAN**, it is not all that easy to learn. A person must learn to think in programming terms. Many of the skills a person needs to program in more difficult languages are also needed to learn **BASIC.**

The New Step By Step teaches programming skills that are common to all small **BASIC**-speaking computers. We feel that a person should learn the basics first and then go on to learn the special features of his or her computer. These features include special graphics, text editing, string manipulations and sound effects.

How To Use The New Step By Step

1. You *cannot* learn to program without actually writing programs. This course will be of little value unless you actually practice what you learn in each lesson.
2. The lessons in the course should be completed in sequence. No lessons should be skipped even if you feel that you already know the material being covered.
3. It is not a good idea to do too much in a session. One complete lesson and a quiz is probably the limit for most people, although some will want to do less.
4. It is best not to let too much time elapse between sessions. Short daily sessions are better than long weekly sessions.
5. The practice sessions are extremely important to effective learning. The student should try all of the material in the practice session.
6. A good way to learn programming is to take someone else's program and make your own modifications to the program to see what happens. This should be done by the student with the programs given in the practice sessions.

You are now ready to begin *The New Step By Step*. If you have an Apple II Plus, insert Disk One into disk drive #1 and turn on the computer.
If you have an Apple II, push the **CTRL** key and the **B** key at the same time. Then type **PR#6**.

A menu will appear on the screen. Select LESSON 1, PART 1.

*Apple is a trademark of Apple Computer Corp.

Lesson 1, Part 1
SUMMARY

Take a look at the computer terminal's keyboard. When you type something into the computer, it is not official until you press the **RETURN** key at the right side of the keyboard.

When you type zero, do not use the letter O. The computer's zero has a slash through it like this: Ø.

When you type a number 1, don't use the letter l. The computer has a number 1 key.

If you make a mistake and wish to change what you have typed, press the backspace key (the key with the arrow pointing to the left.)

The following is an example of what computer memory looks like if it is printed out.

```
ØFØØ 21 ØØ 45 4D 11 ØØ ØE CD 1F ØF B9 CA Ø8 ØF 77 ØØ
ØF1Ø 8Ø 47 23 1B 7A B3 CA 29 ØF CD 1F ØF C3 ØF ØF DB
ØF2Ø ØØ E6 4Ø CA 1F ØF DB Ø1 C9 AF B8 C2 2F ØF 76 FB
ØF32Ø C3 3Ø ØF ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
```

The following is an example of what a machine-language program looks like:

```
FFØØ D8
FFØ1 58
FFØ2 AØ 7F
FFØ4 8C 12 DØ
FFØ7 A9 A7
FFØ9 8D 11 DØ
FFØC 8D 13 DØ
FFØF C9 DF
FF11 FØ 13
FF13 C9 9B
FF15 FØ Ø3
FF17 C8
FF18 1Ø ØF
FF1A A9 DC
```

Interpreter: The interpreter is a machine-language program. Its job is to translate **BASIC** code into computer-understandable machine language and to execute the results of the translation.

BASIC stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode.

BASIC words resemble English words. The following are a few common ones:

PRINT STOP RUN NEW SAVE LOAD INPUT GOTO
READ DATA LET FOR NEXT RETURN RESTORE IF...THEN

A syntax error will result if the interpreter does not recognize a word or character that should be a **BASIC** word or character. A misspelled BASIC word is a very common syntax error.

To use the **PRINT** statements:

1. Type the word **PRINT.**
2. Type the characters you want displayed enclosed in quotation marks.
3. Press the **RETURN** key.

Now, any characters inside the quotation marks will be displayed on the screen.

Lesson 1, Part 1
PRACTICE PROBLEMS

Here are some things to practice with your computer. See what will happen when you type the following commands into the computer. Note that the quotation marks are over the 2. Push the shift key to type quotation marks. Press the **RETURN** key after each command.

```
PRINT "HELLO"
PRINT 'HELLO'
PRINT HELLO
PRINT "HELLO
PRINT "HELLO
PRINT "!??#"
```

Press RETURN after each line.

Note: On some computers this will not produce an error message. But be sure to type the second quotation mark or you will have trouble with more complicated commands.

If you make a mistake in typing a line, you can correct the error by using the ← key. Press the ← key to move back on the line as many spaces as it takes to remove the typing error. Then type in the correct matter. Try it now:

Type: PRINT "HELO THERE
Now press the ← key 7 times
Now finish typing the line so it reads:
PRINT "HELLO THERE"

Try these commands. If you make any errors in typing, use the ← key to correct them.

```
PRINT "YOUR NAME"
PRINT ""YOUR NAME""
PRINT "MY NAME IS "BOB""
PRINT "MY NAME IS 'BOB'"
```

Now practice on your own, using the **PRINT** command. Do not use numbers at this time. When you are ready, load Part 2 of Lesson 1 into the computer by entering the command:

RUN LESSON 1 PART 2

Lesson 1, Part 2
SUMMARY

Arithmetic using **PRINT**:

You enter: PRINT 10 + 6
Computer prints: 16

Inside quotes, numbers are treated just as if they were letters:

You enter PRINT "8 + 9"
Computer prints: 8 + 9

Order of arithmetic:

first done is:	*	—	multiplication
and	/	—	division
next done is:	+	—	addition
and	–	—	subtraction

Example: 10 – 50 / 2 + 8 * 3
step 1: 10 – 25 + 8 * 3
step 2: 10 – 25 + 24
step 3: –15 + 24
step 4: 9

When parentheses are used, work enclosed in parentheses is done first.

Example: 3 * (2 + 5)
step 1: 3 * 7
step 2: 21

When parentheses are used:

First: multiplication and division inside parentheses is done
Second: addition and subtraction inside parentheses is done
Third: multiplication and division outside parentheses is done
Fourth: addition and subtraction outside parentheses is done

Example: (1 + 4 * 3) + 2 * (3 + 3)
step 1: (1 + 12) + 2 * (3 + 3)
step 2: (13) + 2 * (3 + 3)
step 3: (13) + 2 * (6)
step 4: 13 + 12
step 5: 25

Lesson 1, Part 2
PRACTICE PROBLEMS

Let's practice typing commands into the computer. Type each of these commands and press the **RETURN** key. See what the computer responds.

```
PRINT 2+6
PRINT 3*9

PRINT 48/6
PRINT 9ØØØ/2
PRINT 5+6–3+7+19
PRINT "3+3"
```

Note: The * key is directly to the right of the Ø key. Press the shift key and hold it while you type *.

Look at each of the examples below and try to find the solution to each one without using the computer but using the computer's rules. Then type the command into the computer, press the **RETURN** key and see how the computer finds the solution.

If you feel that you understand the rules very well, try the last few examples. If you can solve them, skip the rest.

```
PRINT 4/2+6
PRINT 3*2+1
PRINT 1Ø/5+4–5
PRINT 8*2–6*2
PRINT (3+6)/(3+3)
PRINT (1Ø*4–4)/(2*8–4)
PRINT 2*(6*2)–2Ø
PRINT 25+(5*5)/5
PRINT (6*4+4)/(3+4)–4
PRINT 1ØØ/(1Ø+1Ø)*(2*4)–1
```

Now RUN QUIZ 1.

Lesson 2, Part 1
SUMMARY

This is a program:

```
1Ø PRINT "A"
2Ø PRINT
3Ø PRINT "B"
```

These lines are *not* a program:

```
PRINT "A"
PRINT
PRINT "B"
```

Statements in a program must have line numbers.

Line numbers can be in any increment, but must not be 64000 or over.

It's a good idea to step line numbers by 10 or so to leave space for added numbers for lines you may have to insert.

The command **RUN** executes a program, beginning with the lowest line number and working towards the highest line number.

RUN does not clear memory of the program just executed.

LIST displays all or part of a program in line number order:

LIST	Lists entire program
LIST 10	Lists line number 10
LIST 1 – 100	Lists line numbers 1 through 100
LIST 100 –	Lists line 100 and all lines after
LIST – 100	Lists first line up to and including 100

Lesson 2, Part 2
SUMMARY

To correct a line in a program:

A. Enter the line number with nothing following. This will remove the entire line. Or,

B. retype the line correctly and enter it.

To remove an entire program:

Enter **NEW**.

To remove part of a program:

Enter **DEL** 5, 10 (This means delete lines 5 through 10.)

END ends a program.

STOP stops a program with a line number break message.

Commands are executed when **RETURN** is pressed.

Program statements are executed only when the program is executed, and have line numbers.

Commands such as **NEW**, **LIST**, are rarely used as program statements.

Lesson 2, Parts 1 and 2
PRACTICE PROBLEMS

Type **NEW** and enter these practice programs:

```
10  PRINT "START"
20  PRINT "MIDDLE"
30  PRINT "STOP"
40  END
```

LIST the program just entered.
RUN the program.
Type **NEW** and then **LIST** the program.

```
10  PRINT "8*5="
20  PRINT 8 * 5
30  STOP
40  END
```

LIST it.
RUN it.
Type **NEW**.

```
5   PRINT "MATH BY 2'S"
10  PRINT "2+2="
20  PRINT 2 + 2
30  PRINT "2-2="
40  PRINT 2 - 2
50  PRINT "2*2="
60  PRINT 2 * 2
70  PRINT "2/2="
80  PRINT 2 / 2
90  PRINT "THAT'S IT"
100  STOP
```

LIST it.
RUN it.
(Do not type **NEW** yet.)

Add the following to the last program without retyping in the above statements.

```
 8  PRINT "ADDING"
25  PRINT "SUBTRACTING"
45  PRINT "MULTIPLYING"
65  PRINT "DIVIDING"
```

LIST it.
RUN it.

Note: Line numbers are typed in increments of 10 to allow room to add more statements between the ones already typed in.

Type **LIST** 60 and see what happens.
Type **LIST** 60 – 100 and see what happens.

RUN QUIZ 2.

Lesson 3, Part 1
SUMMARY

A variable is a piece of memory used to hold a value.

A variable may be labeled by:

A. A letter from A through Z.
B. Double letters from AA through ZZ in any combination (AZ; PQ.)
C. A letter followed by a single-digit number from 0 through 9.

The **LET** instruction sets a variable to any number you desire. The actual word **LET** may be omitted. For example:

```
LET Z=1
Y=5
```

PRINTing a variable gives you the value of its contents and doesn't change the value of the variable.

A variable may be used in arithmetic just like the number it contains.

A. naming it and setting it to a new number: LET Y=20
B. letting it equal itself plus a number: LET X=X+8
C. letting it equal itself plus another variable: LET X=X+Y
D. letting it equal another variable: LET X=Y
E. letting it equal the result of arithmetic between other variables or numbers: LET Z=X+Y; LET Z=1+2+3

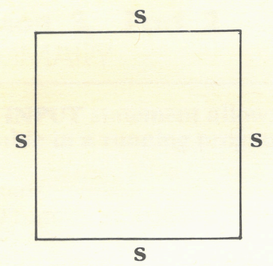
A variable named for the first time in a program is first set to zero automatically.

Lesson 3, Part 1
PRACTICE PROBLEMS

The following are simple programs using variables. Enter each program and **RUN** it. Make the suggested changes and **RUN** it again. Type **NEW** before you go on to the next program.

Do as many of these programs as you like until you feel comfortable with variables.

Program to find the perimeter (distance around) a square.



S represents Side

P represents Perimeter

Perimeter = Side+Side+Side+Side

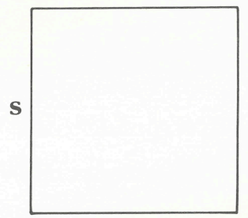
```
10  LET S = 5
20  LET P = S + S + S + S
70  PRINT P
80  END
```

Add the statements with arrows to make an expanded program with callouts:

```
   10  LET S = 5
   20  LET P = S + S + S + S
→  30  PRINT "PERIMETER OF A SQUARE"

→  40  PRINT "SIDE="
→  50  PRINT S
→  60  PRINT "PERIMETER="
   70  PRINT P
   80  END
```

In the above program, change line 10 and find the perimeter of a square with sides of 12; sides of 15; sides of some other number.

Program to find the area of a square.

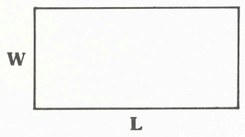


Area = Side × Side

```
10  LET S = 14
20  LET A = S * S
30  PRINT "AREA OF A SQUARE"
40  PRINT "SIDE="
50  PRINT S
60  PRINT "AREA="
70  PRINT A
80  END
```

Change line 10 and find the areas of other squares.

Program to find the area of a rectangle.




```
10  LET L = 3
20  LET W = 2
30  LET A = L * W
40  PRINT "AREA OF A RECTANGLE"
50  PRINT
60  PRINT "LENGTH="
70  PRINT L
80  PRINT "WIDTH="
90  PRINT W
100  PRINT "AREA="
110  PRINT A
120  END
```

Change lines 10 and 20 and find a few other areas.

Program to convert kilometers to miles.

1 Kilometer = .621 Miles


```
10  LET K = 100 ⟵ converts 100k into miles
20  LET M = .621 * K
30  PRINT "KILOMETERS="
40  PRINT K
50  PRINT "MILES="
60  PRINT M
70  END
```

Change line 10 to convert 60 kilometers to miles.

Lesson 3, Part 2
SUMMARY

The **INPUT** statement allows information to be inserted by the user into a variable in a running program.

Formats:

Program statement: 10 INPUT S
Computer prints: ?
You enter: 3
Computer makes S equal 3

Program statement: 10 INPUT X,Y
Computer prints: ?
You enter: 3,5
Computer makes X equal 3 and Y equal 5

Program statement: 10 INPUT Y
Computer prints: ?
You enter: C
Computer prints: ?REENTER
?
You enter: 3
Computer makes Y equal 3

Program statement: 10 INPUT X,Y
Computer prints: ?
You enter: 3
Computer makes X equal 3
Computer prints: ??
You enter: 5
Computer makes Y equal 5

Lesson 3, Part 2
PRACTICE PROBLEMS

Enter the following programs to see how the imput statement works. When you think you understand the input statement, take the Quiz for Lesson 3.

```
5  PRINT
10  PRINT "PROGRAM TO ADD ANY TWO
     NUMBERS"
20  PRINT "INPUT TWO NUMBERS"
30  INPUT A, B
40  PRINT "THE ANSWER IS"
50  PRINT A + B
60  END
```

```
10  PRINT "PROGRAM TO CONVERT GAL
     LONS TO LITERS"
20  PRINT "INPUT NUMBER OF GALLON
     S"
30  INPUT G
40  LET L = 16.907 * G
50  PRINT "NUMBER IN LITERS:"
60  PRINT L
70  END
```

```
10  PRINT
20  PRINT "PROGRAM TO CONVERT FEE
     T TO METERS"
30  PRINT "HOW MANY FEET?"
40  INPUT F
50  LET M = .3048 * F
60  PRINT "CONVERTS TO THE FOLLOW
     ING IN METERS:"
70  PRINT M
80  END
```

As practice, write your own programs to do the following:

A. subtract two numbers
B. multiply two numbers
C. divide two numbers
D. add three numbers
E. multiply three numbers

RUN QUIZ 3.

Lesson 4, Part 1 SUMMARY

The statement **GOTO** followed by a line number will pass control to the line number following **GOTO**.

GOTO can create an infinite loop within a program. To break the loop, hold down the key marked **CTRL** while pressing the key for letter **C** and a **STOP** will be generated for you.

Example:

```
10  PRINT "SAMPLE PROGRAM"
20  PRINT "INPUT A NUMBER"
30  INPUT A
40  PRINT "THE SQUARE IS"
50  PRINT A * A
60  GOTO 20
```

loop

IF. . .THEN

IF checks for a condition. If the condition is true, everything in the line following **THEN** is executed. If the condition is not met, the remainder of the line is ignored.

Illustration:

```
IFA = 3Ø THEN STOP
```

Condition — **What happens or where computer goes if condition occurs.**

IF. . .THEN can be used to print different words depending on a certain condition. For example:

```
10  INPUT A
20  IF A = 1 THEN  PRINT "YES"
30  IF A = 2 THEN  PRINT "NO"
40  IF A = 3 THEN  PRINT "MAYBE"
50  END
```

The **IF. . .THEN** statement will stop a repeating program. This program will stop when we input −1.

```
10  PRINT "PROGRAM TO ADD 2 NUMBE
     RS"
20  PRINT "INPUT 2 NUMBERS"
30  INPUT A, B
40  PRINT A + B
50  IF A =  - 1 THEN  END
60  GOTO 20
```

Note the various ways one can enter the same statement:

1Ø IF A = 1 THEN GOTO 5Ø

1Ø IF A = 1 THEN 5Ø

1Ø IF A = 1 GOTO 5Ø

You enter: 1Ø GO TO 5Ø
Computer lists: 1Ø GOTO 5Ø

You enter: 1Ø G OTO 5Ø
Computer lists: 1Ø GOTO 5Ø

Lesson 4, Part 1
PRACTICE PROBLEMS

Run the following program and then try to stop it. You can always turn off the computer if you forget how to break a loop.

```
10  LET A = 10
20  PRINT A
30  GOTO 10
40  END
```

See what happens when this program is run:

```
10  GOTO 20
20  GOTO 10
30  END
```

Try this program:

```
5  PRINT "TYPE A NUMBER"
10  INPUT A
20  IF A = 1 THEN  PRINT "YES"
30  IF A = 2 THEN  PRINT "NO"
40  IF A = 3 THEN  PRINT "MAYBE"
50  IF A = 4 THEN  STOP
60  GOTO 5
```

Lesson 4, Part 2
SUMMARY

IF. . .THEN and **GOTO** can be used together to send control to various parts of a program depending on a certain condition. For example:

```
10  INPUT X
20  IF X = 1 THEN  GOTO 50
30  IF X = 2 THEN  GOTO 60
40  STOP
50  PRINT X
60  PRINT "END"
70  END
```

Summary of signs:

X=Y	X equals Y
X<Y	X is less than Y
X>Y	X is greater than Y
X>=Y	X is greater than or equal to Y
X<=Y	X is less than or equal to Y
X<>Y	X is not equal to Y

Lesson 4, Part 2
PRACTICE PROBLEMS

ENTER and **RUN** this program:

```
5  LET N = 5
10  PRINT "THIS IS A NUMBER GUESS
    ING GAME"
20  PRINT "TRY TO GUESS A NUMBER
    FROM 1 TO 10."
30  PRINT "YOUR GUESS"
40  INPUT G
50  IF G = N THEN 100
60  IF G <  > N THEN  PRINT "NO,
    GUESS AGAIN"
70  GOTO 20
100  PRINT "THAT'S IT!"
110  END
```

ENTER and **RUN** this program:

```
10  PRINT "THIS IS A NUMBER GUESS
    ING GAME"
20  PRINT "FOR TWO PEOPLE.  ONE PE
    RSON "
30  PRINT "INPUTS A SECRET NUMBER
     BETWEEN 1"
40  PRINT "AND 100 AND CLEARS THE
    SCREEN. "
50  PRINT "THE OTHER PERSON TRIES
     TO GUESS"
60  PRINT "THE NUMBER.  THE PROGRA
    M SUPPLIES"
70  PRINT "HINTS. "
80  PRINT "INPUT THE SECRET NUMBE
    R"
90  INPUT N
100  PRINT "START GUESSING. . . "
110  INPUT G
120  IF G = N THEN 200
130  IF G > N THEN  PRINT "TOO HI
    GH.  GUESS LOWER. "
140  IF G < N THEN  PRINT "TOO LO
    W.  GUESS HIGHER"
150  GOTO 110
200  PRINT "THAT'S IT!"
```

Lesson 5, Part 1
SUMMARY

Counters

When the program begins, A is set equal to zero. Then the program adds 1 to A. Line 25 checks to see whether A equals 5. Since it doesn't, the program loops back to line 10 and adds 1 to A again. It continues in this manner until A = 5.

```
5  LET A = 0
10  LET A = A + 1
20  PRINT A
25  IF A = 5 THEN  END
30  GOTO 10
```

Counter

New line

REM

Tells BASIC interpreter to ignore this line.

Label for program

```
10  REM  THIS IS A SIMPLE PROGRAM
20  PRINT 2 + 2
30  END
]RUN
4
```

COMPUTER PRINTS

Lesson 5, Part 1
PRACTICE PROBLEMS

See what happens when this program is run:

```
10  LET A = A + 1
20  PRINT A
30  GOTO 10
```

Watch how this program stops itself:


```
10  LET A = A + 1
20  PRINT A
30  IF A = 20 THEN  PRINT "WE'RE
     HALF WAY"
40  IF A = 40 THEN  PRINT "THAT'S
     IT"
50  IF A = 40 THEN  END
60  GOTO 10
```

Practice with the following programs:

```
10  PRINT "THIS IS A NUMBER GUESS
     ING GAME"
20  PRINT "FOR TWO PEOPLE.  ONE PE
     RSON "
30  PRINT "INPUTS A SECRET NUMBER
     BETWEEN 1"
40  PRINT "AND 100 AND CLEARS THE
     SCREEN. "
50  PRINT "THE OTHER PERSON TRIES
     TO GUESS"
60  PRINT "THE NUMBER.  THE PROGRA
     M SUPPLIES"
70  PRINT "HINTS. "
80  PRINT "INPUT THE SECRET NUMBE
     R"
90  INPUT N
91  PRINT
92  PRINT
93  PRINT
94  PRINT
95  PRINT
96  PRINT
97  PRINT
98  PRINT
99  PRINT
100  PRINT
101  PRINT
102  PRINT
103  PRINT
104  PRINT
106  PRINT
107  PRINT
108  PRINT
109  PRINT
110  PRINT
111  PRINT
112  PRINT
113  PRINT
114  PRINT "START GUESSING. . . "
115  INPUT G
116  REM  RIGHT GUESS
120  IF G = N.THEN 200
```

continued on next page

```
125  REM  WRONG GUESS
130  IF G > N THEN  PRINT "TOO HI
     GH.  GUESS LOWER."
140  IF G < N THEN  PRINT "TOO LO
     W.  GUESS HIGHER"
145  REM  ERROR COUNT
150  IF G <  > N THEN  LET E = E +
     1
180  GOTO 115
200  REM  CORRECT GUESS
210  PRINT "THAT'S IT!"
220  PRINT "NUMBER OF ERRORS:"
230  PRINT E
235  LET E = 0
240  PRINT "DO YOU WANT TO GO AGA
     IN?"
242  PRINT "(1=YES , 0=NO)"
250  INPUT A
260  IF A = 1 THEN  GOTO 80
270  END
```

```
10  REM  PROGRAM TO COMPUTE BANK
     INTEREST
20  PRINT
30  PRINT "THIS IS A PROGRAM TO C
     OMPUTE COMPOUND"
40  PRINT "INTEREST.  YOU INPUT Y
     OUR STARTING."
50  PRINT "PRINCIPAL, INTEREST RA
     TE, AND NO. OF"
60  PRINT "YEARS THE PRINCIPAL IS
      IN THE ACCOUNT."
70  PRINT "WE WILL GIVE YOU YOUR
     BANK BALANCE FOR"
80  PRINT "THE END OF THE PERIOD.
     "
90  PRINT "INPUT PRINCIPAL, INTERE
     ST (DECIMAL), AND NUMBER OF
     YEARS."
92  PRINT "THE THREE NUMBERS SHOU
     LD BE SEPARATED BY COMMAS."
95  REM  P IS PRINCIPAL, R IS INT
     EREST RATE
96  REM  AND Y IS NUMBER OF YEARS

100  INPUT P,R,Y
110  REM  PRINCIPAL WITH INTEREST
      ADDED TO FIND
115  REM  NEW PRINCIPAL
120  LET P = P + P * R
130  REM  COUNTER FOR NUMBER OF Y
     EARS
```

continued on next page


```
135  LET A = A + 1
140  IF A = Y THEN 200
150  GOTO 120
200  PRINT " YOUR PRINCIPAL PLUS
     INTEREST FOR THAT"
210  PRINT "PERIOD OF TIME IS:"
220  PRINT P
230  END
```

Lesson 5, Part 2
SUMMARY

Library Functions:

X = INT (Y)	Integer function: makes X equal to Y stripped of its decimals
X = ABS (Y)	Absolute function: makes X equal to Y stripped of its plus or minus sign
X = RND(1)	Random function: makes X equal to a random number between 0 and 1.
X = SQR (Y)	Square root function: makes X equal to the number which, multiplied by itself, equals Y

To get random integers between 1 and 50:

```
10 X = RND (1)          10 X = INT (50 * RND (1)) + 1
20 X = X * 50
30 X = INT (X)
40 X = X + 1
```

Nested Parentheses:

```
X = (5 + (2 * (3 + 4)))
X = (5 + (2 * (7     )))
X = (5 + (2 *  7     ))
X = (5 + (14         ))
X = (5 + 14          )
X = 5  + 14
X = 19
```

Other library functions:

X = SIN (Y)	X equals Sine of Y in radians
X = COS (Y)	X equals Cosine of Y in radians
X = TAN (Y)	X equals Tangent of Y in radians
X = ATN (Y)	X equals Arctangent of Y in radians
X = LOG (Y)	X equals Natural Logarithm of Y

Lesson 5, Part 2
PRACTICE PROBLEMS

ENTER and **RUN** the following programs:

```
10  REM  COUNTING BY FIVE THE HAR
    D WAY
20 X = 0
30 X = X + 1
40  IF X = 100 THEN  END
50  IF X / 5 =  INT (X / 5) THEN
    PRINT "THE COUNTER IS UP TO
    "
60  IF X / 5 =  INT (X / 5) THEN
    PRINT X
70  PRINT
80  GOTO 30
```

```
10  REM  HEADS OR TAILS PROGRAM
20  LET H = 0
30  LET T = 0
40  PRINT "THE COMPUTER WILL FLIP
    A COIN 10 TIMES"
50  LET N =  RND (1)
55  LET N =  INT (2 * N)
60  IF N = 0 THEN  PRINT "HEADS"
70  IF N = 0 THEN H = H + 1
80  IF N = 1 THEN  PRINT "TAILS"
90  IF N = 1 THEN T = T + 1
100  IF H + T = 10 THEN 120
110  GOTO 50
120  PRINT "TOTAL HEADS"
130  PRINT H
140  PRINT "TOTAL TAILS"
150  PRINT T
160  END
```

```
10  REM  NUMBER GUESSING GAMES WI
    TH RANDOM NUMBERS
20  PRINT "THE COMPUTER WILL THIN
    K UP A NUMBER"
30  PRINT "BETWEEN 0 AND 100, YOU
    TRY TO GUESS IT."
40  REM  COMPUTER THINKS UP NUMBE
    R
50  LET N =  RND (1)
60  LET N =  INT (100 * N)
70  PRINT "OK, TAKE A GUESS."
80  INPUT G
```

continued on next page


```
90  REM  RIGHT GUESS
100  IF G = N THEN 200
120  IF G > N THEN  PRINT "TOO HI
    GH.  GUESS LOWER"
130  IF G < N THEN  PRINT "TOO LO
    W.  GUESS HIGHER"
140  REM  ERROR COUNT
150  LET E = E + 1
160  GOTO 80
200  REM  CORRECT GUESS
210  PRINT "CORRECT"
220  PRINT "NUMBER OF ERRORS:"
230  PRINT E
235  LET E = 3
240  PRINT "DO YOU WANT TO GO AGA
    IN? (1=YES, 0=NO)"
250  INPUT A
260  IF A = 1 THEN 50
270  END
```

RUN QUIZ 5

Lesson 6, Part 1
SUMMARY

There are a number of commands in Applesoft which affect the way in which words and symbols are printed on the screen.

To clear the screen, you can use the **HOME** instruction.

```
10  PRINT "OLD PAGE"
20  HOME        <-- Clears the screen.
30  PRINT "NEW PAGE"
40  END
```

The **PRINT** command can be used to keep things from looking crowded on the screen.

```
10  PRINT "A"
20  PRINT      <-- Causes a blank line to be printed between A and B.
30  PRINT "B"
40  END

]RUN
A

B
```

When the **BASIC** interpreter sees a comma in a **PRINT** statement, it places the material after the comma in the next "column" on the screen. (Of course, the comma can't be enclosed in quotation marks. Then, the computer would print the comma.)

```
10 A = 1
20 B = 2
30 C = 3
40  PRINT A, B, C
50  END

RUN

1     2     3 ← computer prints
```

When a semicolon appears in a **PRINT** statement, no spaces are left between the words or numbers being printed.

```
10 A = 1
20 B = 2
30  PRINT "A"; "B"  Prints the letters.
40  PRINT A; B  Prints the values of A and B.
50  END

RUN

AB
 12
```

Semicolons and commas tell the computer to continue printing on the same line instead of moving down a line.

```
10  PRINT "BACK"
20  PRINT "UP"
30  END

RUN

BACK

UP ← computer prints
```

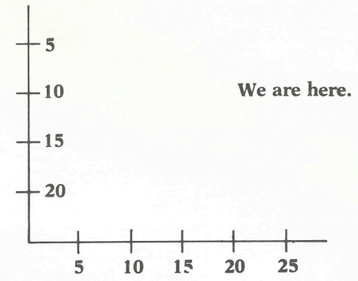
```
10  PRINT "BACK";  Semicolon
20  PRINT "UP"
30  END

RUN

BACKUP ← computer prints
```

The **TAB** command or statement works the way a **TAB**ulator key works on a typewriter. It tells the computer on which column of the screen to print.

```
PRINT TAB (15); "BEGIN"
          ↓
Tells the computer to go to
column 15.
          ↓
..............BEGIN ← Computer prints
```


To print lines in a specific location on the screen, use **VTAB** and **HTAB**. **VTAB** tells the computer what line to print on no matter what line it is currently on. **HTAB** tells the computer how far to go across the screen before printing.

```
10  HOME
20  VTAB 10
30  HTAB 20
40  PRINT "WE ARE HERE"
50  END
```



Lesson 6, Part 1
PRACTICE PROBLEMS

RUN these **PRINT** commands and see what happens.

```
PRINT 5,6,7
PRINT 5;6;7
PRINT TAB(20);"B"
PRINT TAB(20);"8+5"
PRINT TAB(20);8+5
PRINT "FIRST";TAB(20);"LAST"
```

```
5  REM  USING VTAB AND HTAB
10  HOME
20  PRINT "WHAT LINE DO YOU WANT
     TO PRINT ON?"
30  INPUT A: IF A > 22 THEN  PRINT
     "TOO LARGE.  TRY AGAIN. "
35  IF A > 22 THEN  GOTO 30
40  PRINT "HOW FAR ACROSS THE SCR
     EEN WILL YOU PRINT?"
50  INPUT B
60  IF B > 40 THEN  PRINT "TOO LA
     RGE.  TRY AGAIN"
65  IF B > 40 THEN  GOTO 50
70  PRINT "WHAT DO YOU WANT PRINT
     ED?"
80  INPUT A$
90  HOME
100  VTAB A
110  HTAB B
120  PRINT A$
```

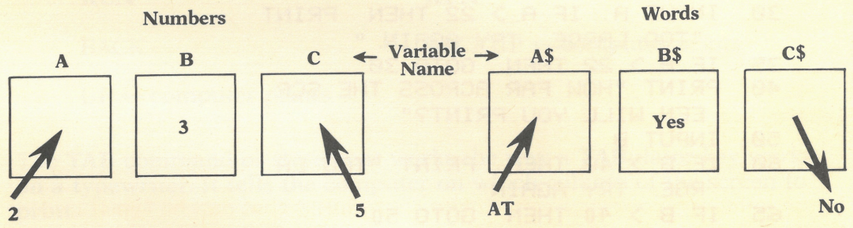
continued on next page

```
130  PRINT
140  PRINT
150  PRINT "DO YOU WANT TO CONTIN
     UE ? (1=YES, 0=NO)"
160  INPUT A$
170  IF A$ = "0" THEN  END
180  GOTO 10
```

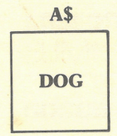
```
5  REM  EXAMPLE OF COLUMNS FORMED
      BY COMMAS
10  HOME
20  PRINT
30  PRINT
40  PRINT "CHART OF SQUARE AND CU
     BE NUMBERS"
50  PRINT
60  PRINT "NUMBER","SQUARE","CUBE
     "
70 N = N + 1
80  PRINT N,N * N,N * N * N
90  IF N = 10 THEN  END
100  GOTO 70
```

Lesson 6, Part 2
SUMMARY

Variables can contain letters or words as well as numbers. A variable which contains numbers can be given any letter name. The same is true for a variable which contains letters, but this name must be followed by a $.



A$ holds strings of characters, so it is called a string variable.



Letters can be placed in string variables much the same way that numbers are placed in regular variables, with one exception. The information going into the string variable must be in quotation marks.

Let A = 8, or A =8 LET A$="YES", or A$="YES"

String variables can be used like any other variables in **PRINT** statements and in **INPUT** statements.

```
10  LET A$ = "STOP"
20  PRINT A$
30  END

RUN

STOP
```

```
10  INPUT A$
20  PRINT A$
30  END

RUN

?STOP ← User imputs

STOP ← Computer prints
```

Almost any character can be put into a string variable, including a space.

```
10 A$ = "THE MARTIAN SAID 'TAKE M
     E TO YOUR LEADER #$%&()*'"
20  PRINT A$
30  END
```

Two strings must match exactly or the computer will decide that they are not the same:

A$="PIN" B$="PINE"
A$<>B$

Lesson 6, Part 2
PRACTICE PROBLEMS

In the following program, experiment with things that can be typed into the string variable from the keyboard.

```
10  INPUT A$
20  PRINT
30  PRINT "THIS IS WHAT YOU TYPED
      INTO THE STRING  VARIABLE."

40  PRINT A$
50  END
```

Try these two programs.

```
10  REM  SILLY ANSWERS TO INPUTS
     PROGRAM
15  HOME
20  PRINT "WHAT IS YOUR NAME?"
30  INPUT A$
40  PRINT
50  PRINT A$;"! WHAT KIND OF NAME
      IS THAT?"
60  PRINT "WHERE DO YOU LIVE?"
70  INPUT B$
80  PRINT B$;"? I HAVE A FRIEND W
     HO LIVES IN ";B$;"."
90  PRINT "OK, "A$;", GO BACK TO
     WHATEVER YOU WERE DOING";
95  PRINT " IN ";B$;"."
100  END
```

```
5  HOME
10  REM  MADLIB
20  PRINT "TYPE IN THE NAME OF A
     PLACE OR A THING:"
30  INPUT A$
40  PRINT "TYPE IN THE NAME OF A
     BODY PART:"
50  INPUT B$
60  HOME
70  PRINT "HERE IS YOUR MADLIB"
80  PRINT
90  PRINT "JACK AND JILL"
100  PRINT "WENT UP THE ";A$
110  PRINT "TO FETCH A PAIL OF WA
     TER."
120  PRINT "JACK FELL DOWN"
130  PRINT "AND BROKE HIS ";B$
140  PRINT "AND JILL CAME TUMBLIN
     G AFTER!"
150  END
```

RUN QUIZ 6

Lesson 7, Part 1
SUMMARY

GOSUB use:

```
10 ...
20 ...
30 ...
40 GOSUB 1000
45 ...
50 ...
55 ...
60 ...
70 ...
80 END
1000  PRINT
1010  PRINT "TYPE 0 AND PRESS THE
      RETURN KEY."
1020  INPUT A
1030  PRINT
1040  RETURN
```

Consequences of missing **END** statements:

```
10 ...
20 ...
30 ...
40 GOSUB 1000
45 ...
50 ...
55 ...
60 ...
70 ...
80 ...
1000  PRINT
1010  PRINT "TYPE 0 AND PRESS THE
      RETURN KEY."
1020  INPUT A
1030  PRINT
1040  RETURN
?RETURN WITHOUT GOSUB IN 1040
```

Consequences of missing **RETURN** statements:

```
10 ...
20 ...
30 ...
40 GOSUB 1000
45 ...
50 ...
55 ...
60 ...
70 ...
80 END
1000  PRINT
1010  PRINT "TYPE 0 AND PRESS THE
      RETURN KEY."
1020  INPUT A
1030  PRINT
2000 ...
2010 ...
2030 ...
2040 ...
```

Lesson 7, Part 1
PRACTICE PROBLEMS

Try these two programs:

```
10  REM  GOSUB DEMONSTRATION
15  HOME
20  PRINT "GO TO SUBROUTINE"
30  GOSUB 1000
40  PRINT "AGAIN"
50  GOSUB 1000
60  PRINT "THIRD TIME"
70  GOSUB 1000
80  STOP
1000  PRINT "HERE'S THE ROUTINE"
1010  PRINT "**********"
1020  PRINT
1030  RETURN
```



```
10  REM  ARITHMETIC PROGRAM WITH
     SUBROUTINE
15  HOME
20  PRINT "WHAT'S YOUR NAME?"
30  INPUT A$
40  PRINT
50  PRINT A$;", DO THE FOLLOWING"

52  PRINT "MULTIPLICATION PROBLEM
    S."
55 E = 0
57  PRINT
60  LET A =  INT (10 *  RND (1))
70  LET B =  INT (10 *  RND (1))
80  PRINT A;"X";B;"=";
90  INPUT C
100  IF C = A * B THEN 140
110  LET E = E + 1
112  IF E = 1 THEN W = W + 1
114  IF E = 3 THEN  PRINT "THE RI
    GHT ANSWER IS ";A * B
115  IF E < 3 THEN  PRINT "WRONG
    TRY AGAIN."
116  IF E = 3 THEN  GOTO 55
117  PRINT
118  GOTO 80
140  GOSUB 1000
150  REM  CHECK TO SEE IF STUDENT
     HAS DONE 10 PROBLEMS."
160  IF W + R = 10 THEN 200
170  GOTO 60
200  PRINT A$;", YOU SCORED ";R;"
     RIGHT OUT OF 10"
210  END
1000  REM  SUBROUTINE
1010  LET Q =  INT (3 *  RND (1))

1020  IF Q = 0 THEN  PRINT "CORRE
    CT"
1030  IF Q = 1 THEN  PRINT "YOU'V
    E GOT IT, ";A$
1040  IF Q = 2 THEN  PRINT "EXCEL
    LENT"
1050  LET R = R + 1
1060  PRINT
1070  RETURN
```

Lesson 7, Part 2
SUMMARY

READ. . .DATA

```
10  READ X
20  PRINT X
30  GOTO 10
40  DATA  1,2,3,4,5,6
RUN
1
2
3
4
5
6
?OUT OF DATA IN 10
```

Computer prints X.

Computer prints error message for your machine.

RESTORE

RESTORE makes the computer use the data again.

```
10  READ Y
20  PRINT Y
30  IF Y =  - 1 THEN  RESTORE
40  GOTO 10
50  DATA  1,2,3,-1
RUN
1
2
3
-1
1
2
3
-1
```

data restored here.

Computer prints

Here the computer reads a number for X and the next one for Y each time:

```
10  READ X,Y
20  PRINT X,Y
30  GOTO 10
40  DATA  1,2,3,4,5,6
RUN
1             2
3             4
5             6
?OUT OF DATA ERROR IN 10
```

Computer prints X, Y each time.

Computer is out of data.


```
10  READ X$
20  PRINT X$
30  GOTO 10
40  DATA  ONE, TWO, THREE, GO

]RUN
ONE
TWO
THREE
GO
?OUT OF DATA ERROR IN 10

10  READ X$
20  PRINT X$
30  DATA  ONE TWO THREE GO

]RUN
ONE TWO THREE GO
```

Lesson 7, Part 2
PRACTICE PROBLEMS

Try the following programs:

```
10  REM  READ. . .DATA PRACTICE
20  READ A
30  PRINT A
40  GOTO 10
50  DATA  5,6,7,8
```

RUN the above program and then add the following line to it:

```
35  IF A = 8 THEN  RESTORE
```

Add the following lines:

```
5  LET B = 0
36  LET B = B + 1
37  IF B = 8 THEN  STOP
```

Add the following line:

```
6  DATA  1,2,3,4
```

Add the following lines:

```
22  READ X,Y
23  PRINT X,Y
```

and delete 35, 36, and 37.

RUN QUIZ 7

Lesson 8, Part 1
SUMMARY

This is an example of a counting loop. For each value of A from 1 to 3, the computer prints "***"

```
10 A = 0
20 A = A + 1
30  PRINT "***"
40  IF A = 3 THEN  END
50  GOTO 20
JRUN
***
***
***
```

A **FOR. . .NEXT** statement performs the same function as a counting loop. The following program will have the same printout as the program in the previous exercise.

```
10  FOR A = 1 TO 3   Sets lower and upper limits for A.
20  PRINT "***"
30  NEXT A           Completes loop
JRUN
***
***
***
```

This is how a **FOR. . .NEXT** loop works:

Give A a value and go on

```
10  FOR A=1  TO  10
20  NEXT  A
```

Go back to Get next A

All statements that are included within the **FOR. . .NEXT** statements are executed each time the loop is made.


```
10  FOR A = 1 TO 2
20 Z = A * A
30  PRINT "A*A =";Z
40 Q = A / Z
50  PRINT "A/Z =";Q
60  PRINT
70  NEXT A
```

Lines 20–60 are executed each time the value of A increases.

```
JRUN
A*A =1
A/Z =1
A*A =4
A/Z =.5
```

It is not necessary to use a variable name with a **FOR. . .NEXT** statement. You just say **NEXT.**

```
10  FOR A = 1 TO 10
20  PRINT A
30  NEXT   Do not need to say NEXT A.
```

Different-sized steps can be used in counting in **FOR. . .NEXT** loops. This program uses steps of 10 in counting from 10 to 50. (Without a step statement, **FOR. . .NEXT** loops always count by one.)

```
10  FOR Y = 10 TO 50 STEP 10
20  PRINT Y
30  NEXT Y
40  END
JRUN
10   ] Steps of 10
20
30
40
50
```

Computer prints

FOR. . .NEXT statements can count backwards as well as forwards. This program counts backwards by 2 each time it performs a loop:

```
10  FOR X = 10 TO 2 STEP  - 2
20  PRINT X;
30  NEXT X
JRUN
108642
```

Lesson 8, Part 1
PRACTICE PROBLEMS

Try these practice problems.

```
10  REM  SIMPLE FOR..NEXT PROGRAM

20  FOR A = 1 TO 10
30  PRINT A
40  NEXT A
50  END
```

Add this line to the above program:

```
35  PRINT A * A
```

Add this line:

```
45  PRINT "THE END"
```

```
10  REM  COUNTDOWN PROGRAM
15  HOME
20  FOR X = 10 TO 0 STEP  - 1
30  PRINT X
40  NEXT X
50  PRINT "*******BOOM*******"
```

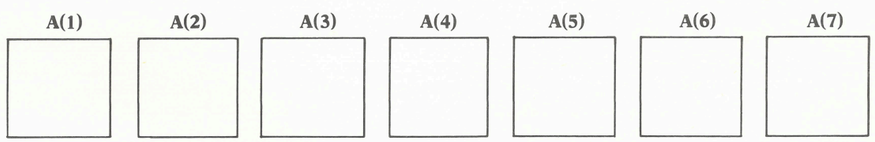
Note: **Clear the screen with HOME.**

Add this to the above program:

```
35  FOR Y = 1 TO 500
36  NEXT Y
```

Lesson 8, Part 2
SUMMARY

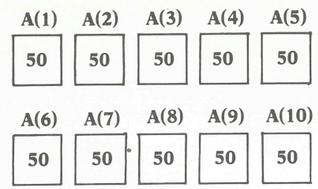
A line of variables is called an **ARRAY**.



Variables in an array all have the same letter name followed by a different number for each variable. The numbers are always in parentheses.

This program will set up an array of 10 variables, each containing the number 50.

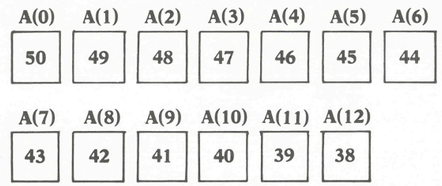
```
10  FOR X = 1 TO 10
20  LET A(X) = 50
30  NEXT X
40  END
```



Before using an array that has more than 11 variables, you must make space in the computer by using a **DIM** statement. If you have less than 11 variables, a **DIM** statement is not necessary.

```
10  DIM A(12)
20  FOR X = 0 TO 12
30  LET A(X) = 50 - X
40  NEXT X
50  END
```

Sets up an array of 13 variables beginning with A(0) and ending with A(12).



Lesson 8, Part 2
PRACTICE PROBLEMS

Try these programs.

```
10  PRINT "THIS IS THE RESERVATIO
    N SYSTEM FOR"
20  PRINT "LOWER SLOBOVIA AIRWAYS
      OUR BIPLANE HAS"
30  PRINT "4 SEATS ON IT.  YOU CAN
    RESERVE FLIGHTS 1"
40  PRINT "TO 7 (SORRY, ALL FLIGH
    TS SECOND-CLASS)."
50  REM  SET UP SEATS ON PLANE
60  FOR X = 1 TO 7
70  LET A(X) = 4
80  NEXT X
90  PRINT "WHAT FLIGHT (1 THRU 7)
    DO YOU WANT?"
```

continued on next page

```
100  INPUT X
110  IF A(X) = 0 THEN  PRINT "SOR
     RY, FLIGHT CLOSED.  TRY ANOTH
     ER..."
120  IF A(X) <  > 0 THEN 140
130  GOTO 90
140  REM  REMOVE SEAT FROM AVAILA
     BILITY
150  LET A(X) = A(X) - 1
160  PRINT "OK TO ISSUE TICKET FO
     R THAT FLIGHT."
170  GOTO 90
```

```
10  REM  RANDOM NUMBERS PUT INTO
     AN ARRAY
20  FOR C = 1 TO 10
30  LET N(C) =  INT ( RND (1) * 1
     00)
40  NEXT
50  REM  PRINT THE ARRAY
60  PRINT "RANDOM ARRAY"
70  PRINT
80  FOR C = 1 TO 10
90  PRINT "N(";C;")=";N(C)
100  NEXT C
```

RUN QUIZ 8

Lesson 9, Part 1
SUMMARY

Programs can be simplified by using nested loops, loops that are within other **FOR. . .NEXT** loops. The Z loop in program B replaces the 5 PRINT statements in Program A.

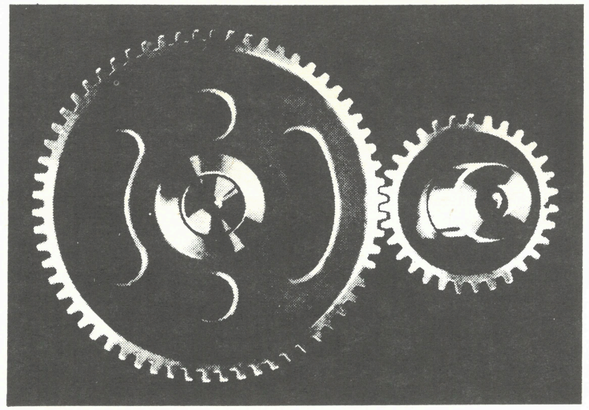
Program A

```
10  FOR A = 1 TO 3
20  PRINT "1****"
30  PRINT "2****"
40  PRINT "3****"
50  PRINT "4****"
60  PRINT "5****"
70  NEXT A
```

Program B

```
10  FOR A = 1 TO 3
20  FOR Z = 1 TO 5
30  PRINT Z;"****"
40  NEXT Z
50  NEXT A
```


Nested loops work a little like these gears. When the large gear turns once, the smaller gear turns several times.



For each time the program moves once through the outside loop, it moves five times through the inside loop.

```
10  FOR A = 1 TO 100
20  FOR B = 1 TO 5
30  NEXT B
40  NEXT A
```

Outside loop

Inside loop

The correct placement of **NEXT** statements is important in **FOR. . .NEXT** loops. In Program A below, the **NEXT** statement for the **Z** loop is out of place. Rather than increasing the value of **Z,** the computer returns to line 10 and increases the value of A each time it gets to line 40. When line 50 is reached, an error message appears.

Program A

```
10  FOR A = 1 TO 4
20  FOR Z = 1 TO 2
30  PRINT Z
40  NEXT A
50  NEXT Z
]RUN
1
1
1
1
?NEXT WITHOUT FOR ERROR IN 50
```

Program B

```
10  FOR A = 1 TO 4
20  FOR Z = 1 TO 3
30  PRINT Z
40  NEXT Z
50  NEXT A
]RUN
1
2
3
1
2
3
1
2
3
1
2
3
```

Correct placement of NEXT statements.

Lesson 9, Part 1
PRACTICE PROBLEMS

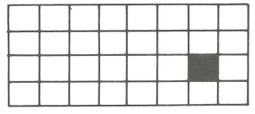
Try these programs.

```
10  FOR P = 1 TO 10
20  PRINT "PROBLEM"
30  FOR N = 1 TO 2
40  PRINT "GIVE ME A NUMBER"
50  INPUT A(N)
60  NEXT N
70  PRINT A(1);"*";A(2);"=";A(1) *
     A(2)
75  PRINT
80  NEXT P
90  END
```

```
10  HOME
20  PRINT "HOW MANY PEOPLE IN YOU
     R FAMILY?"
30  INPUT N
40  FOR C = 1 TO N
50  PRINT "HOW OLD IS PERSON ";C;
     " ?"
60  INPUT A(C)
70  NEXT C
80  PRINT "HERE'S A GRAPH OF YOUR
      AGES."
90  FOR C = 1 TO N
100  PRINT "PERSON ";C
110  FOR X = 1 TO A(C)
120  PRINT "*";
130  NEXT X
140  PRINT
150  NEXT C
```

Lesson 9, Part 2
SUMMARY

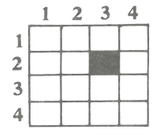
The screen of a computer is divided into squares, somewhat like this section of graph paper.



Square is on.

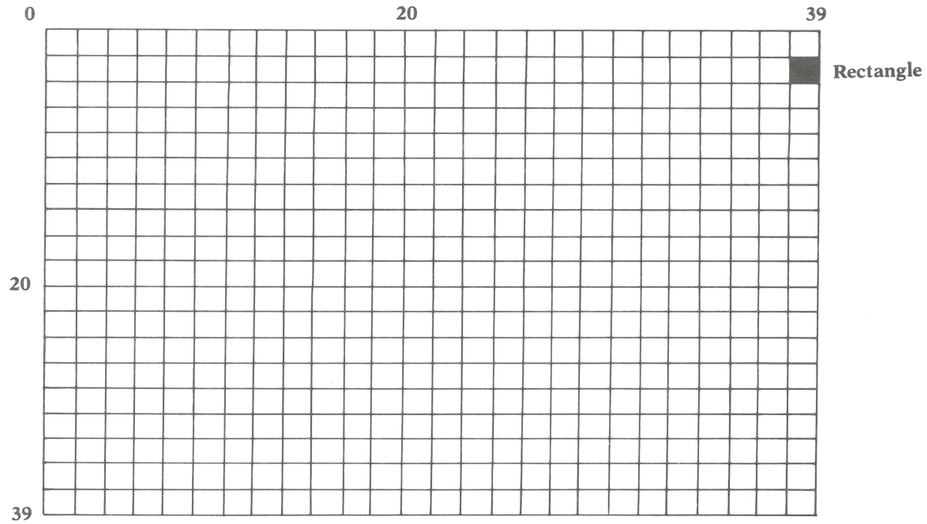
Each square can be turned *on* or *off*. Turning squares *on* or *off* makes patterns we call computer graphics.

Each of these squares has a certain location on the screen. Here is a location 3 across and 2 down.



In **GRAPHICS**, the screen of the computer is divided into 40 columns and 40 rows. The columns are numbered from 0 to 39 starting at the left and increasing to the right. The rows are also numbered from 0 to 39 from the top to the bottom of the screen.

Because the screen is wider than it is long, the squares formed by the intersecting rows and columns are actually rectangles.



Screen In Graphics

Each rectangle represents a certain location on the screen. The rectangle shown in the previous illustration has a location of 39 across the screen and 1 down. Locations are shown with the horizontal number first and the vertical one, second. A comma separates the two numbers. For example:

39, 1
horizontal, vertical

To use Apple Graphics, you must first give the graphics command, which is **GR**.

```
10  GR  <-- Sets up the screen for graphics.
20  COLOR= 6
30  PLOT 10,10
40  END
```

The second step is to specify one of the 16 colors in graphics. These are:

0	Black	8	Brown
1	Magenta	9	Orange
2	Dark Blue	10	Grey
3	Purple	11	Pink
4	Dark Green	12	Green
5	Grey	13	Yellow
6	Medium Blue	14	Aqua
7	Light Blue	15	White

Colors in Apple Graphics.

To specify a color, you use the **COLOR** command.

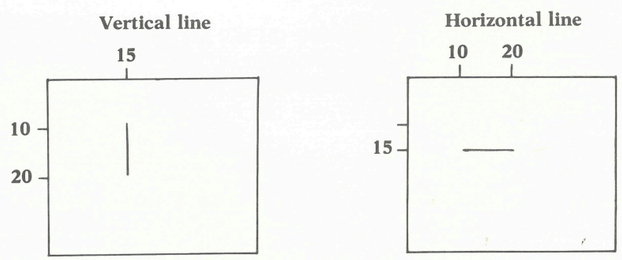
```
10  GR
20  COLOR= 6  Specifies color 6, medium blue.
30  PLOT 10,10
40  END
```

The third step in graphics is to turn on a location. The instruction here is **PLOT**.

```
10  GR
20  FOR C = 1 TO 15
30  COLOR= C
40  PLOT 20,10  <-- Plots a location 20 across and 10 down on the screen.
45  FOR X = 1 TO 500
47  NEXT X
50  NEXT C
```

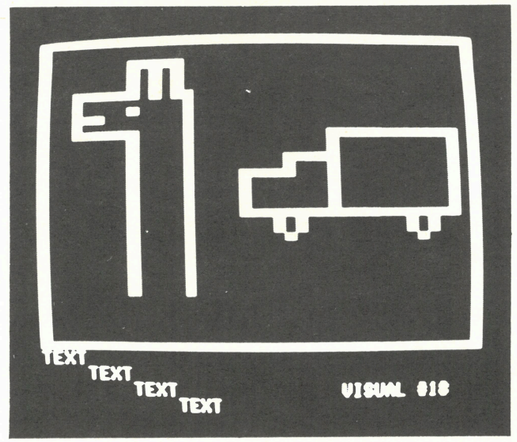
This program changes the color of the location 20,10 each time the value of **C** changes.

In addition to plotting individual points, lines can be drawn on the screen. The commands used are **VLIN** for vertical lines and **HLIN** for horizontal lines.



The first two numbers in each command show the end points of the line. The third shows where the line is positioned, either across the screen for vertical lines or down the screen for horizontal lines.

The graphics command divides the screen into 2 parts. The 4 lines on the bottom of the screen are used for text. The portion above the text is used for graphics. Text and graphics cannot appear on the same line.



Graphics

To get back into the regular print mode, you need the **TEXT** command, followed by the **HOME** command.

```
10  GR
20  COLOR= 6
30  PLOT 5,5
40  FOR T = T TO 1000  Timing loop
50  NEXT T
60  TEXT  gets rid of graphics
70  HOME  clears the screen.
80  PRINT "WE ARE NOW BACK IN REG
     ULAR PRINT MODE."
90  END
```

If you forget to use the command **TEXT**, line 80 will produce illegible lines on the screen when this program is run.

Lesson 9, Part 2
PRACTICE PROBLEMS

Enter this program and try various values of **X** and **Y** to see where the locations are.

```
5  GR
10  COLOR= 15
15  PRINT "TYPE VALUES FOR X AND
     Y WITH A COMMA"
17  PRINT "IN BETWEEN. "
20  INPUT X,Y
30  PLOT X,Y
40  PRINT "PRESS RETURN TO CONTIN
     UE. "
50  INPUT A$
60  GOTO 15
```

Enter this program and run it to see what happens.

```
5  GR
10  COLOR= 15
20 X =  INT ( RND (1) * 40)
30 Y =  INT ( RND (1) * 40)
40  PLOT X,Y
50  GOTO 20
```

Change line 10 to the following and try to predict what will happen:

```
1Ø COLOR=INT (16*RND (1))
```

```
1  REM  PROGRAM TO DRAW HORIZONTA
     L LINES
5  GR
10  FOR C = 1 TO 15
20  COLOR= C
30  FOR X = 0 TO 39 STEP 3
40  HLIN 0,39 AT X
50  NEXT X
60  NEXT C
70  END
```

Change the above program to draw vertical lines.

Then change the program to draw the lines and erase them before drawing new ones. (Hint: COLOR=Ø turns a location off.)

If you add the following line to the above program, you will make a grid:

```
45 VLIN Ø,39 AT X
```



```
10  GR
15  COLOR= 13
20  FOR X = 0 TO 39
30  HLIN 0,39 AT X
35  NEXT X
40  REM   NOW WE DRAW THE FRAME
45  COLOR= 2
50  HLIN 0,39 AT 0
55  HLIN 0,39 AT 39
60  VLIN 0,39 AT 0
65  VLIN 0,39 AT 39
```

Run the above program. Then add the following:

```
70  REM   MAKES MOVING DOT BY TUR
     NING LOCATION
71  REM   ON AND OFF
75  FOR X = 0 TO 39
80  COLOR= 0
90  PLOT X,20
100  REM   TIMING LOOP
105  FOR D = 1 TO 59
110  NEXT D
120  COLOR= 13
130  PLOT X,20
140  NEXT X
150  REM  80 AND 90 TURN DOT OFF
151  REM  120 AND 130 TURN IT BAC
     K ON
```

Run the above program and then add the following:

```
155  FOR X = 39 TO 0 STEP  - 1
160  COLOR= 0
170  PLOT X,20
180  FOR D = 1 TO 50
190  NEXT D
200  COLOR= 13
210  PLOT X,20
220  NEXT X
```

RUN QUIZ 9

Lesson 10, Part 1
SUMMARY

To create a multi-statement line, separate each statement with a colon. For example.

```
1Ø PRINT A : GOTO 1Ø
```

GOSUB: In a multi-statement line, a **RETURN** will go back to the next colon.

IF. . .THEN: In a multi-statement line, if the condition is not met, the entire remainder of the line is ignored.

REM: In a multi-statement line, everything after **REM** for the remainder of the line is not executed.

The question mark (?) is the abbreviation for **PRINT** while entering a program.

Lesson 10, Part 1
PRACTICE PROBLEM

Enter the following program "as is." Then see how much shorter you can make it by using multiple statement lines.

```
1  HOME
5  PRINT "NUMBERS AND SQUARES"
10  GOSUB 1000
20  READ Y
30  PRINT Y;
50  REM  COMPUTE SQUARES
70  PRINT " SQUARES TO:";
80  PRINT Y * Y
85  GOSUB 1000
90  IF Y <  > 10 THEN  GOTO 20
200  PRINT "THAT'S IT."
210  END
1000  PRINT "****************"
1020  RETURN
2000  DATA  1,2,3,4,5,6,7,8,9,10
```

Lesson 10, Part 2
SUMMARY

Exponents:

```
2∧1 =     2 =     2
2∧2 =     2*2 =     4
2∧3 =     2*2*2 =     8
2∧4 =     2*2*2*2 =     16
```

NEW ORDER OF ARITHMETIC:

∧	—	exponents
*	—	multiplication
/	—	division
+	—	addition
−	—	subtraction

Scientific notation:

```
2000  =  2  *  1000
2000  =  2  *  10∧4
2000  =  2E        4
2000  =  2E  04
2000  =  2E+04

2345  =  2.345  *  1000
2345  =  2.345  *  10∧4
2345  =  2.345  E  4
2345  =  2.345  E  +  04

6.5E+06  =6.5  *  10∧6
6.5E+06  =6.5  *  1000000
6.5E+06  =6500000

.00000007  =  7  *  .00000001
.00000007  =  7  *  10∧−8
.00000007  =  7E−08

.00023  =  2.3  *  10∧−4
.00023  =  2.3E−04

2.34E−09  =  2.34  *  10∧−9
2.34E−09  =  2.34  *  .000000001
2.34E−09  =  .00000000234
```

Lesson 10, Part 2
PRACTICE PROBLEMS

Enter this program. Then try the numbers below as input and see how the computer responds.

```
10  PRINT "ENTER A NUMBER.": INPUT
    A
20  PRINT A
30  GOTO 10
```

SAMPLE INPUT:

```
1000000
1,000,003
2330000000000000
.01
.003
.1234567
.001234567
1.00000056
.0000000333333
```

Try other numbers.

Then change the program so you input two numbers and multiply them. See when you get scientific notation for numbers.

Then, divide the two numbers. Try to get answers in scientific notation.

RUN QUIZ 10

WRITTEN QUESTIONS FOR LESSONS 1–5 QUIZ

23. Write a program that will print from 1 to 10 and end the program when it hits 10.

24. Write a statement that will produce a random number from 1 to 100.

25. Write a statement that will put −7.3 in a box called A.

WRITTEN QUESTIONS FOR LESSON 6–10 QUIZ

11. Write a statement that will put the following string in B$:
    YOU WERE THERE

12. Write a program that will imput your name and then print it out as follows:

    ?JOHN
    HELLO JOHN!

13. Write a program using a **FOR-NEXT** loop that will count backwards from 10 to 1 and print out the results.

14. Write a program that will count from 1 to 5 using **READ. . .DATA** statements.

15. Write a program that will assign the value of 1 to each variable in an array from A(1) to A(5).

(answers on reverse)

ANSWERS FOR LESSONS 1–5 QUIZ

23. Here is the most elementary way to answer the question.

```
1Ø REM PROGRAM TO COUNT TO 1Ø AND STOP
2Ø LET X = Ø
3Ø LET X = X + 1
4Ø PRINT X
5Ø IF X = 1Ø THEN STOP
6Ø GOTO 3Ø
```

24. 1Ø PRINT INT (RND (1)*1ØØ) + 1

25. 1Ø LET A = −7.3 or 1Ø A = −7.3

ANSWERS TO LESSONS 6–10 QUIZ

11. 1Ø LET B$ = "YOU WERE THERE"
    or
    1Ø B$ = "YOU WERE THERE"

12. 1Ø INPUT N$
    2Ø PRINT "HELLO" ;N$; "!"

13. 1Ø FOR X = 1Ø TO 1 STEP −1
    2Ø PRINT X
    3Ø NEXT X
    4Ø END

14. There are several ways to do this—here is the way we did it:

```
1Ø DATA 1, 2, 3, 4, 5
2Ø READ C
3Ø PRINT C
4Ø IF C = 5 THEN STOP
5Ø GOTO 2Ø
```

15. 1Ø FOR X = 1 TO 5
    2Ø LET A(X) = 1
    3Ø NEXT X